

Create Performance Task

Row 2 - Data Abstraction


Instructions

1. You can only use this version if you can write on a pdf, otherwise please use the ppt version.
2. Read the criteria for this row on slides 3 - 4.
3. For each response (*slides 5 – 14*):
 - a. Underline any code, phrases or sentences that meet any of the criteria.
 - b. Tick (✓) any criteria that are satisfied and cross (✗) any criteria that are not satisfied.
 - c. Write a 1 in the bottom right corner if the response gets the mark for this row (*all criteria have been satisfied*), otherwise write a 0.
4. When finished, save this presentation as a pdf with 2 slides to a page and submit this file.

Scoring Criteria

- includes two program code segments:
 - one that shows how data has been stored in this list (*or other collection type*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

Decision Rules

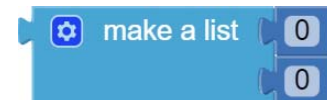
- Consider ONLY written response 3b when scoring this point.
- Requirements for program code segments:
 - The written response must include two clearly distinguishable program code segments, but these segments may be disjointed code segments or two parts of a contiguous code segment.
 - If the written response includes more than two code segments, use the first two code segments to determine whether or not the point is earned.
- Do NOT award a point if any one or more of the following is true:
 - The list is a one-element list. 
 - The use of the list does not assist in fulfilling the program's purpose.

The list must not be empty or only have 1 element.

e.g. Neither of the following are acceptable:



One way round this would be to write:



However, there would normally be a point where actual required data is added to the list and another point where the list is subsequently used, so I suggest using those code segments instead.

a

```
if (turn % 2 != 0)
{
    grid[column_check_gridposition1][column_check_gridposition2] = 'X';
    turn_absolute = 'O';
    current_turn = 'X';
}
else
{
    grid[column_check_gridposition1][column_check_gridposition2] = 'O';
    turn_absolute = 'X';
    current_turn = 'O';
}
```

3.b.iii.

The list (2-dimensional array) used in the code above is given the name "grid." In the second image, "grid" is accessed within a function as an argument and given the local name "current_grid."

3.b.iv.

The data contained in this list represents the "status" of each position on the connect-four game board. The array is initialized with the dimensions of 6 x 7, with each position representing the respective position on the game board. The "status" is denoted by an "X" (piece belongs to player X), an "O" (piece belongs to player O), or a " " (position is empty).

```
// "current_grid" is local variable of "grid"
if (current_grid[i][j] == current_grid[i + 1][j] &&
    current_grid[i][j] == current_grid[i + 2][j] &&
    current_grid[i][j] == current_grid[i + 3][j])
{
    return 1;
}
else if (current_grid[i][j] == current_grid[i][j + 1] &&
    current_grid[i][j] == current_grid[i][j + 2] &&
    current_grid[i][j] == current_grid[i][j + 3])
{
    return 1;
}
else if (current_grid[i][j] == current_grid[i + 1][j + 1] &&
    current_grid[i][j] == current_grid[i + 2][j + 2] &&
    current_grid[i][j] == current_grid[i + 3][j + 3])
{
    return 1;
}
else if (current_grid[i][j] == current_grid[i + 1][j - 1] &&
    current_grid[i][j] == current_grid[i + 2][j - 2] &&
    current_grid[i][j] == current_grid[i + 3][j - 3])
{
    return 1;
}
else
{
    return 2;
}
```

- includes two program code segments:
 - one that shows how data has been stored in this list (>1 element).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

b

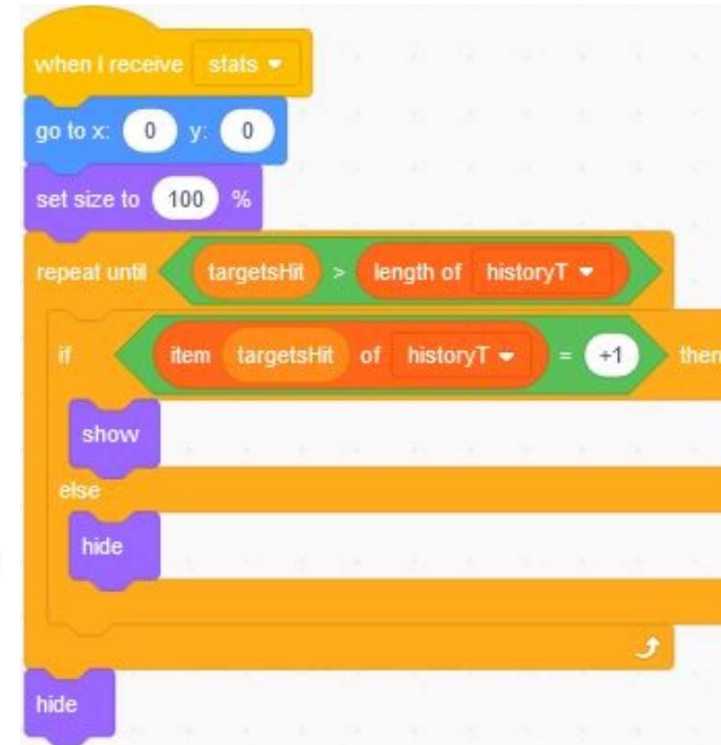


3.b.iii.

This list named "historyT" is a list of all targets that were clicked by representing them by their corresponding values from three different targets: +1, +2, and +3. The code shown above corresponds to one of the targets with a value of +1 when clicked.

3.b.iv.

The data contained in this list represents the specific targets that were hit in chronological order.



- includes two program code segments:
 - one that shows how data has been stored in this list (*>1 element*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

C

```
score_point_list = [(360,40), (315,175), (345,310), (30,100), (135,310), (285,310)]
```

```
for i in range(6):  
    if (score_point_list[i][0] == character.get_x()) and (score_point_list[i][1] == character.get_y()):  
        add(sign_two)  
        add(alert)  
        add(alert_part_two)  
        sign_two.set_position((get_width()/2)-150, (get_height()/2)  
        alert.set_position(90, (get_height()/2)-75)  
        alert_part_two.set_position((get_width()/2)-120, (get_height()/2)  
        add_mouse_click_handler(restarts)  
        score_list.append(1)
```

3.b.iii.

The list being processed is named `score_point_list`.

3.b.iv.

The `score_point_list` contains the coordinates of all the stars and mystery blocks which allows the program to identify where these items that give the user points are located.

- includes two program code segments:
 - one that shows how data has been stored in this list (*>1 element*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

d

```
4 //Here are the lists of movies that a person can get based on the
5 var youthComedy = ["Despicable Me", "Cars", "Rio", "Trolls", "The
6 var youthAction = ["Spy Kids", "The Incredibles", "Big Hero 6",
7 var teenComedy = ["10 Things I Hate About You", "Mean Girls", "
8 var teenAction = ["The Hunger Games", "Divergent", "The Maze Ru
9 var adultComedy = ["13 Going on 30", "The Breakfast Club", "Leg
10 var adultAction = ["Charlie's Angels", "The Fast and The Furious

51
52 //This function tells what movie options the user gets based on their age and genre choice.
53 function getMovie(genre) {
54   if (age <= 12) {
55     if (genre == "Action") {
56       for (var ya = 0; ya < 3; ya++) {
57         appendItem(movie, youthAction[randomNumber(0, youthAction.length-1)]);
58       }
59       setScreen(▼ "movieOutputScreen");
60       setText(▼ "movieOutputText", (movieText + movie));
61     } else if (genre == "Comedy") {
```

3.b.iii.

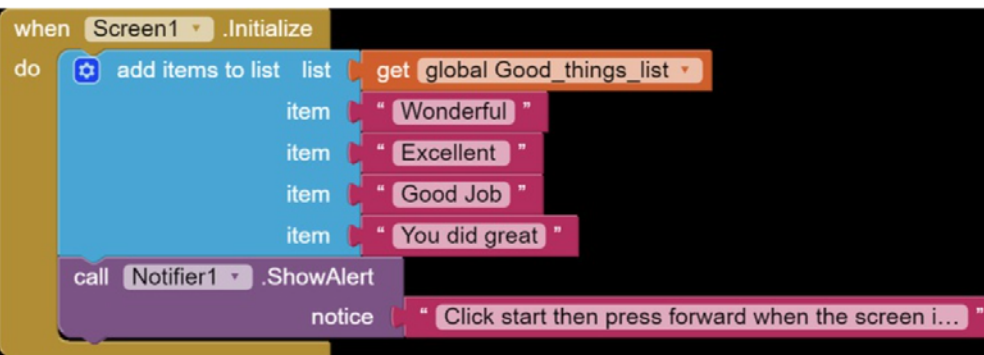
In the image above, you are able to see the youthAction and the youthComedy list being used in two different loops. In the video, youthComedy, teenAction, and adultComedy were being utilized.

3.b.iv.

All of my lists store the names of different comedy or action movies that a user can be suggested. These movies are divided into age-appropriate as well as genre-appropriate categories. From here, these movies are randomly selected and suggested to the user (when the user inputs age and genre choice).

- includes two program code segments:
 - one that shows how data has been stored in this list (*>1 element*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?



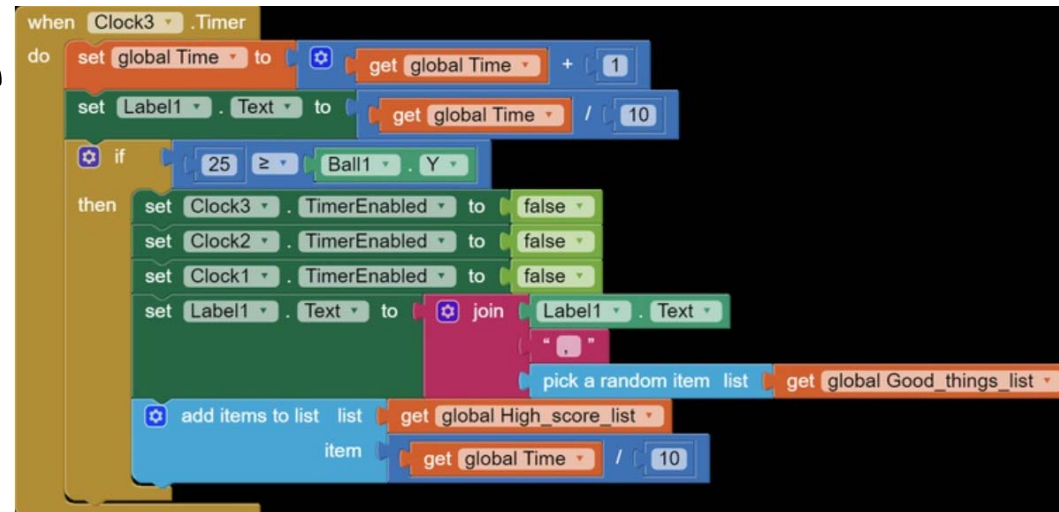
3.b.iii.

The name of the list is Global Good_things_list

3.b.iv.

The data in this list represents the good word that is said after your time is given to you in the app.

e



?

- includes two program code segments:
 - one that shows how data has been stored in this list (*>1 element*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

```
appendItem(hours, getNumber("inputHours"));
```

f

```
function findTotal(hours) {  
    var total = 0;  
    for (var i = 0; i < hours.length; i++) {  
        total = total + hours[i];  
        setText("totaloutput", "You've listened to " + (total + " hours of music this week"));  
        if (total > 27) {  
            setText("iftotal>27", "Wow! You've already listened to more music than the average person  
per week! ");  
        }  
    }  
}
```

3.b.iii.

The name of the list is *hours* }

3.b.iv.

The data in the hours list represents each hour amount the user inputs by using the add button.

- includes two program code segments:
 - one that shows how data has been stored in this list (*>1 element*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

g

```
//The sources of the song lyric and movie quotes
var quoteList = getColumn("quoteList", "quote");
var quoteType = getColumn("quoteList", "type");
var type;
onEvent("movieButton", "click", function( ) {
    type = "movie line";
    filter(type);
}
```

```
for (var i = 0; i < quoteList.length; i++) {
    if (quoteType[i] == userChoice) {
        appendItem(filteredList, quoteList[i]);
        output = output + quoteList[i] + "\n";
    }
}
setText("outputText", output);
```

3.b.iii.

The name of the list being processed is quoteType.

3.b.iv.

The data in this list stores the type of each of my quotes (either song lyric or movie line). This information is used to filter my list and sort the user's choice into an empty list to be output.

- includes two program code segments:
 - one that shows how data has been stored in this list (>1 element).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

h

```
var states = getColumn(▼"Female State Legislators", ▼"State");
```

```
function findMost() {  
  var enter = getText("pickCategory");  
  var category = getColumn("Female State Legislators", enter);  
  var max = 0;  
  var mostName;  
  for(var i = 0; i < category.length; i++){  
    if(category[i] > max) {  
      max = category[i];  
      mostName = states[i];  
    }  
  }  
  setText("results2", ("The state with the most " + enter) + " is " + mostName);  
}
```

3.b.iii.

The name of the list being processed in this response is called Female State Legislators. This list contains all of the information in order to provide the user with the answer they are looking for based on what they input into the app.

3.b.iv.

The Female State Legislators represent the list from which the answers are deduced. The answers are based on what the user chooses throughout the app. For example, if the user selects State - Most Representatives and in the dropdown box chooses Female State Independents, then the answer the user will receive in the text area is

The state with the most Female State Independents is Alaska

. In other words, the name of the list, Female State Legislators, contains the data used to provide the user with the necessary information.

- includes two program code segments:
 - one that shows how data has been stored in this list (>1 element).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

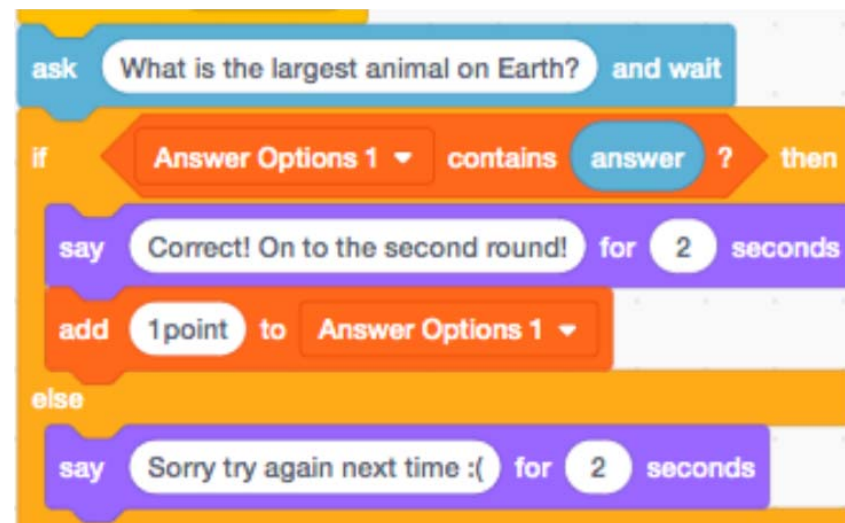


3.b.iii.

In the photo above these variables are being added to the "Answers Option 1" list.

3.b.iv.

The data within the "Answer Option 1" list is being used to allow only those answers to modify the points awarded to the user after they input one of the answers that are on that same list. If the same answer is not input when the question is asked they are not rewarded the point and it affects their final score.



- includes two program code segments:
 - one that shows how data has been stored in this list (*>1 element*).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?

```
listA = ["Math", "History", "Art", "English", "Science"]
listB = ["Earth", "Venus", "Mercury", "Neptune", "Saturn"]
listC = ["Sand", "Ocean", "BeachHome", "VollyBallCourt", "Hotel"]
listD = ["Room", "Bow", "Kitchen", "DiningRoom", "GangWay"]
listE = ["Kitchen", "Tables", "Curtains", "lobby"]

list_list = [listA, listB, listC, listD, listE]
def Mainprogram ():
    strikes = 0
    points = 0
    repeat1 = 0
```

```
listA = ["Math", "History", "Art", "English", "Science"]
listB = ["Earth", "Venus", "Mercury", "Neptune", "Saturn"]
listC = ["Sand", "Ocean", "BeachHome", "VollyBallCourt", "Hotel"]
listD = ["Room", "Bow", "Kitchen", "DiningRoom", "GangWay"]
listE = ["Kitchen", "Tables", "Curtains", "lobby"]

list_list = [listA, listB, listC, listD, listE]
def Mainprogram ():
    strikes = 0
    points = 0
    repeat1 = 0
```

3.b.iii.

There are multiple lists names that are all correlated to different sections of the program. Each list within the program will be chosen once one of the options are chosen and it will be used for that option only. Each of the lists will only pop be used for the correlated option and its category.

3.b.iv.

The data within the lists will give you objects, buildings, planets, for the scenery that matches with the option you chose. Each of the list will be used to gain a points to be able to win the program, five points will win the game.

location city

- includes two program code segments:
 - one that shows how data has been stored in this list (>1 element).
 - one that shows the data in this same list being used as part of fulfilling the program's purpose.
- identifies the name of the variable representing the list being used in this response.
- describes what the data contained in this list is representing in the program.

?